KBCom

By: Edward W. Kuns

A World of Difference...

"KBCom is the reason i've felt no need to write an OS-9 Telecommunication program, despite urging from numerous people"

Rick Adams
CoCo Sig / Database Manager - Delphi
Author of DelphiTerm

"KBCom is without a doubt the best VT100 emulation program for OS9 Level 2 I've seen, and I've seen a bunch. It will become the standard by which all others are judged."

Zack Sessions
Color Systems
Author of WP Shell

Distributed by Second City Software

KALA SOFTWARE
P.O. BOX 39073
GREENSBORO, NC 27438
Formerly Second City Software

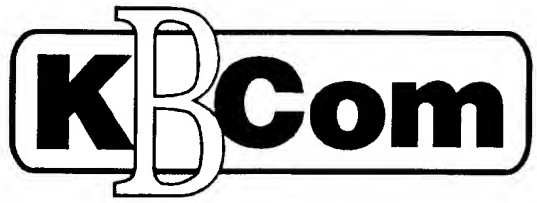


TABLE OF CONTENTS

KBCom

What is K-Bare Com?

K-Bare Com (which will be referred to as KBCom in the manual) is a flexible terminal program that allows you to call and log into other computers. Some of the features of KBCom include:

- VT100, VT52, ANSI, OS-9 and CRT terminal emulation
- Powerful macros which can cause KBCom to wait for a string, prompt the user for input, or pause for a specified amount of time
- Hot-Keys for special functions
- The ability to 'extend' KBCom by calling other programs, or 'extensions'
- All alphanumeric keys are user definable as macros, hot-keys, or extensions
- An Alarm
- A timer that you can use to keep track of how long you've been logged on
- Conference mode to edit text locally and then send it all at once
- Logging the session to a file
- Copy a snapshot of the screen into a file
- Execute any OS-9 command, sending the output through the modem
- Full control over all RS-232 and window parameters
- XON/XOFF flow control
- ASCII, XModem, YModem, YModem batch, and Kermit file transfers

The following equipment is required:

OS-9 Level II Operating System • CoCo 3 with at least 256k memory
One disk drive • Deluxe RS-232 Pak or equivalent
300, 1200, or 2400 bps Modem

To enhance the usefulness of the program, the following are recommended but not necessary:

RGB monitor • RAM Drive • Second Disk Drive or a Hard Disk Drive

History:

At work, I use EDT and TPU and other full screen editors which require sophisticated VT100 terminal emulation. After waiting several months and finding that no-one wrote the program I needed, I decided to do it myself! When naming the program, I came up with two possible names: K-Term, in the grand Color Computer tradition of naming the program after yourself, and Bare-Com, because my initial intention was a program that emulated the VT100 and did nothing else. I finally decided to merge the names together.

Acknowledgements:

I would like to acknowledge the help of many people, without whom KBCom would not exist today. First, I would like to thank Tim Kientzle. Without his kick in the rear to get me motivated, I would never have started this project. (I wanted someone else to write it for me!) His many suggestions and expert help have been invaluable, and helped prevent me from falling on my face when I might have without the benefit of his expertise. James Connolly, Jon Howell, and Greg Law also contributed a flood of ideas just when I thought my own might run out. Thanks are also due to KBCom's many beta testers, who are too numerous to list here. Finally, without the expert help I found on Delphi and on the CoCo bitnet mailing list, KBCom wouldn't have grown nearly so fast.

INSTALLATION:

Before you do anything else, back up the original KBCom disk and place the original in a safe place. Use the backup disk to install KBCom.

KBCom requires certain files to be copied to a specific directory or subdirectory on the working disk. To make this process easier, an install program is supplied that will do this automatically. The install program will also creating the required directories as needed and allow you to customize KBCom to your needs.

By default, parameter files and the KBCom special fonts are stored in the directory /DD/COM/KBCOM. When prompted by the install program for the parameter file directory, press ENTER to use this directory. Alternatively, you can specify the directory to be used; the install program will customize KBCom to reflect this choice. For example, many people prefer to use the directory /DD/SYS or /DD/SYS/KBCOM. Since this directory may be different for each user, I will refer to this directory as the "parameter file directory" from now on.

Before installing KBCom, first make sure the destination disk is formatted with at least 400 free sectors. KBCom doesn't need to be installed onto a boot disk, but you may wish to install it on one. To save time and allow for single drive installations,

the install program loads `copy` and `makdir` before starting the actual installation.

When you are ready to install KBCom, insert the backup KBCom disk into drive 0 and type the following at the OS-9 prompt:

```
chd /d0
chx /d0/cmds
install
```

The first question install asks is the number of disk drives you are using to install KBCom. If you are using two floppy drives, insert the destination disk into drive 1 and enter 2 at the prompt. If you are installing KBCom on a hard disk, enter 2 at the prompt. If you have only one floppy drive without a hard disk, enter 1 at the prompt and you will be prompted to swap disks.

Next, install asks which directory you want the parameter files copied to. If you press ENTER, KBCom will use /DD/COM/KBCOM as the default directory. If you prefer to use another directory, type the name of the directory and press ENTER. If the parameter file directory does not already exist on the destination disk, it will be created on the destination disk. The device name in the parameter file directory specification is only used when KBCom is running, not during installation.

INSTALLING KBCOM BY HAND:

You may install KBCom yourself, if you wish. First create the parameter file directory on the desired disk. Next, copy the files KBCom, XYModem, and Kermit from the CMDS directory on the backup disk to the CMDS directory on the destination disk. Finally, copy all files from the directory COM/KBCOM on the backup disk to the parameter file directory on the destination disk. All files with the extension .kbc are sample parameter files; you may leave them off the destination disk. If you are using a parameter file directory which is different from the default choice of /DD/COM/KBCOM then you will have to patch KBCom to use your choice. Use a disk zapper and change the string /DD/COM/KBCOM to the pathname of your choice; the string occurs only once. Your choice MUST contain less than 40 characters, and end with a NULL (character 0). DO NOT FORGET TO VERIFY KBCOM WHEN YOU FINISH.

IMPORTANT NOTE:

DO NOT merge any files with KBCom or any of its utilities! KBCom fits very tightly in its 64k memory map. If you merge any modules with KBCom, it will probably become too large to fit into memory along with the data it needs. If this happens, KBCom can not run! Thanks to Mike Knudsen for discovering this problem with his program, ***UltiMusE III.***

REQUIRED PATCHES:

If you are using shell+ then you may have to patch it. Shell+ is an enhanced version of the stock Tandy shell which provides many new features. One such feature is that all programs run from the shell will run with a minimum of 31 pages (7936 bytes) of memory. However, KBCom (and many other large programs) will not be able to allocate necessary extra memory, and thus will not run properly.

The modpatch script in PATCHES/shellplus.mod on the installation disk will disable this feature so programs will be run with the default amount of memory (or the amount specified on the command line with the # shell command). To apply this patch, first load modpatch into memory by:

```
load modpatch
```

Next, place the installation disk in drive 0 and execute the following commands:

```
chd /d0/patches
modpatch < shellplus.mod
```

These commands only change the version of shell in memory. You may wish to copy shellplus.mod to your boot disk and run modpatch in your startup file. A more permanent method is to use a disk zapper and apply the patches by hand directly to shell on disk.

OPTIONAL PATCHES:

You have the option of installing two patches in your `OS9Boot` file. These patches are described below:

AciaPak:

The stock Tandy ACIAPAK driver has an 80-character input buffer that can overflow in 1/3 of a second! What this means in real-life terms is that if KBCom misses four consecutive time slices (which can happen if you are running programs in other windows) while you are recieving at 2400 bps, the buffer will overflow. KBCom can use the stock ACIAPAK robustly, assuming you are not running many background tasks with high priorities; however, screen updates may be choppy. The included modpatch script, which I got permission to distribute with KBCom, will increase ACIAPAK's input buffer to 256 bytes. The author of the patch is unknown; I found the patch in the OS-9 SIG on CompuServe.

I highly recommend applying this patch if you are using Tandy's ACIAPAK driver. Although the patch is not required, it helps prevent characters from being lost. Also, when ACIAPAK's buffer fills, the driver may crash -- effectively destroying

the connection. There is little that any terminal program can do to prevent this, and the only way to recover from such a crash is to kill KBCom and start over.

WindInt or GrfInt:

There is a bug in the stock Tandy WindInt and GrfInt modules. The `SS.FBRgs` system call (which returns the palette registers used for the foreground, background, and border colors) will occasionally return the wrong palette registers. Modpatch scripts are provided for both WindInt and GrfInt, so you can patch the window driver you are using. (WindInt is the window interface which is part of the Multi-Vue package. GrfInt is the interface that comes with OS-9 Level II.) Without this patch, KBCom may set the wrong foreground, background, and border colors when exiting.

INSTALLING THE PATCHES:

The easiest way to install these patches is to boot OS-9, format a new disk, and then apply the patches in memory. First load modpatch and cobbler into memory by typing the following commands:

```
load modpatch cobbler
```

Now, put the KBCom installation disk in drive 0 and type the following commands:

```
chd /d0/patches
chx /d0/cmds
modpatch  <  acia_patch.mod
```

If you are using GrfInt, type the following command:

```
modpatch < grfint_fix.mod
```

For the WindInt module, use the following command:

```
modpatch < windint_fix.mod
```

If you only have one floppy drive, then you will have to work directly on your boot disk. You may wish to make a backup of your boot disk before you continue in case something goes wrong. You will need to have at least 250 sectors free in the largest block on your boot disk, maybe more depending on your `OS9Boot`. When you are ready, place your boot disk in drive 0 and type:

```
cobbler /d0
```

If cobbler fails, it will tell you. Since you are using cobbler on a disk which is already

a boot disk, it will warn you that track 34 has not been rewritten. This is normal and does not represent a problem.

If you have two floppy drives, you can create a new boot disk. This saves you the risk of damaging your old boot disk. Insert your freshly formatted disk in drive 1 and your old boot disk in drive 0. Save the modified boot from memory to disk and copy the rest of the old boot disk to the new one by typing the following commands:

```
chd /d0
chx /d0/cmds
cobbler /d1
dsave -s32 /d0 /d1 ! shell
```

RUNNING KBCOM:

Run KBCom as follows:

```
kbcom {rs232_device} {parameter_file} {-a}
```

Where `rs232_device` is the device you want to use for rs232 I/O and `parameter_file` is the file you want parameters to be loaded from upon startup. The arguments can be in any order and none of the arguments are required. KBCom defaults to /t2 for the rs232 device and 'default.kbc' for the parameter file. The '-a' option causes any AUTOSTART macro in the parameter file to be ignored. Without this option, the AUTOSTART macro will be executed (if one is in the parameter file) immediately after parameters are loaded.

Unless the parameter file filename is a full pathname (one beginning with a slash, '/'), the parameter file will be searched for within the default parameter file directory, which is usually /DD/COM/KBCOM. If you specify no extension, then .kbc will be assumed.

Note that KBCom currently demands that it be run on an 80-by-24 text or graphics screen. If run on a smaller screen, it will print an error message and exit. The reason for this is that VT100, VT52, and ANSI emulation requires an 80-by-24 screen to work properly.

KBCom also expects certain utilities to be either in memory or in the current execution directory. Shell is used to execute extensions and for the on-line command. Load is used to preload modules for faster execution and unlink is used to remove these utilities from memory. Finally, merge is used to load the fonts when you enable the USE_FONTS flag.

During initializatioin, KBCom automatically raises its priority by three. If started with a priority of 128 (the default used by OS-9), KBCom will increase its priority

to 131. This helps guarantee that no characters are lost when you run programs in other windows. When no characters are being send or recieved KBCom puts itself to sleep; KBCom only takes advantage of its higher priority when it really needs it.

USING KBCOM FROM GSHELL (Multi-Vue):

You can call a computer by clicking on an icon and never touching the keyboard! All you have to do to accomplish this is to define an AUTOSTART macro in the parameter file that will call a BBS or on-line service and then log on. Then, using Multi-Vue, double-click on the directory with the parameter files in it and double-click on the desired parameter file icon. The AUTOSTART macro will take over, and you will be logged in!

You don't have to do anything other than create a functional AUTOSTART macro. Thus, if each system you log into has its own parameter file with an AUTOSTART macro, you have an instant auto-dialer within Multi-Vue!

THE KEYBOARD:

All 128 ASCII character codes can be sent from KBCom, including all 33 control codes. The control codes are generated by pressing the following keys:

You press	to generate	Sometimes called
CTRL-@	0	NULL or ^Space
CTRL-A to CTRL-Z	1 - 26	
CTRL-BREAK	27	Escape, ^[, META
CTRL-Up arrow	28	^\, ^?, or ^_
CTRL-Left arrow	29	^]
CTRL-Right arrow	30	^^, ^~, or ^'
CTRL-Down arrow	31	^_ or ^?
CTRL-;	127	Del, Rubout, or ^?

Some ASCII characters that are not on the CoCo's keyboard can be generated by OS-9. These include:

You press	to generate	You press	to generate
SHIFT-@	' (Grave accent)	CTRL-,	{
CTRL-1	|	CTRL-.	}
CTRL-3	~	CTRL-/	\
CTRL-8	[	CTRL--	_ (Underscore)
CTRL-9	]		

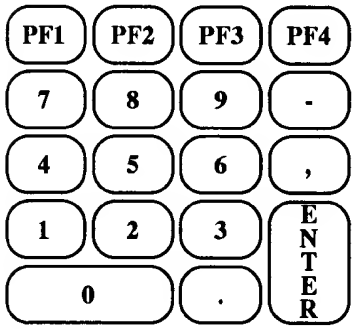
Note: CTRL-A means hold down the CTRL key and press the A key. A shorthand notation often used to indicate control characters is the caret (^). For example, you may see CTRL-A annotated as ^A. This manual uses the notation SHIFT-A, CTRL-A, ALT-A.

The only characters in the tables above that are translated within KBCom are the control-arrow sequences. Also, a bug in OS-9 causes CTRL-@ keypresses to be lost most of the time, so you may have to press CTRL-@ several times to generate a NULL character. The control sequences in the last column (for example, ^SPACE) indicate the various ways these keys are referred to and generated on different computer systems. A grave accent cannot be displayed on an OS-9 text screen. Instead, OS-9 displays a grave accent (') on a text screen as a single quote (').

The arrow keys can send the standard OS-9 arrow characters, or they can send whatever character(s) are required by the current terminal emulation. The parameter ARROW_KEYS determines what characters are sent by the arrow keys. You always have access to both sets of arrow keys, however. If the arrow keys are set to OS-9 arrow keys, then you can send an emulation (VT100, VT52, or ANSI) arrow key by pressing SHIFT-arrow. This also works for the reverse case, when the arrow keys are set to Emulation arrow keys.

If OS-9 arrow keys are chosen, you have an additional choice (still with the parameter ARROW_KEYS) for the left arrow (or backarrow). Many systems use the delete character to backspace. For these systems, you may want to select OS-9/Del for the arrow keys so they send normal OS-9 characters for all arrows except the backarrow, which sends a delete character. If you have Emulation arrow keys selected, then SHIFT-<LEFT ARROW> will always send the normal OS-9 left arrow character.

The only keypads currently supported are the VT100, VT52, and ANSI keypads, which look like this



Generate the PF keys as follows:
PF1 press F1
PF2 press F2
PF3 press SHIFT-F1
PF4 press SHIFT-F2

and the rest by pressing ALT and one of the non-PF keypad keys. In other words, press ALT-, to generate a keypad comma.

EMULATIONS:

There are a number of "standard" computer terminals, each one with its own set of control sequences to position the cursor, clear the screen, and perform other actions the terminal is capable of. One of the most widely used standards is the DEC VT100. ANSI emulation is a variant and subset of VT100 which adds a couple commands to control screen colors.

These standards are useful when you wish to take advantage of full-screen editors and other programs that use special terminal control sequences. By relying on the terminal to do some of the work the application is able to work much faster.

KBCom currently supports VT52, VT100, ANSI, OS-9 and CRT terminal emulation. Support for VT52, VT100, and ANSI includes the application keypad when the VT_KEYPAD parameter is enabled. When VT_KEYPAD is disabled, the ALT-0 through ALT-9 keys serve as extra macros. The layout of the application keypad is discussed in the section on the keyboard. VT100 and ANSI emulations also support bold, blinking, and inverse video. The VT100 special graphics character set, double-width characters, and double-width, double-height characters are supported if KBCom is run on a graphics screen. Due to limitations of OS-9 on the Color Computer, 132-column mode is not supported and blinking characters are not supported on a graphics screen. The fonts for the VT100 special graphics and double-width fonts are defined in the file `standard.fnt` in the parameter file directory. When you enable the USE_FONTS flag, the fonts are loaded into memory. You don't need to and shouldn't preload the fonts. The fonts are automatically removed from memory when you disable USE_FONTS and/or when the program ends. The VT100 fonts are in font group 199.

To use these special fonts, you must run KBCom in a graphics window and enable USE_FONTS. When you load the fonts, there are several possible error conditions. If you attempt to turn fonts on when KBCom is running on a text screen, Use_Fonts is turned off automatically. The other possible errors are: "Can't find font file" -- the file `standard.fnt` isn't in the parameter file directory, or the parameter file directory cannot be found. "Can't find normal font" -- the font file was found, but missing essential fonts. Use_Fonts is turned off if any error is encountered.

Note: the font file contains special fonts. You cannot use just any font file when USE_FONTS is enabled. Use the program KB_Font to convert your favorite fonts into fonts usable with KBCom's VT100 mode. If you wish to run KBCom on a graphics window without turning USE_FONTS on, then you can, of course, use any font. Simply load and select that font before running KBCom.

UPLOADING AND DOWNLOADING:

KBCom does not have any file transfer protocals built-in. Considering the wide variety of protocals in common use, it would be impossible to make a useful number of them all fit into a program that would still run on the Color Computer! Thus, KBCom is designed to use separate file transfer programs. This makes KBCom flexible; anyone may use their favorite method of transfering files provided a program that performs the transfer is available.

XYModem, which comes with KBCom, provides XModem, YModem, and YModem batch transfers. The sample parameter files which come with KBCom demonstrate how to use XYModem and kermit to transfer files. For more information, refer to the documentation for these programs (and any other external file transfer programs you wish to use with KBCom), the description of extensions in this manual, and the sample parameter files.

The version of kermit distributed with KBCom is public domain. Source, full documentation, and future updates can be found on Delphi, CIS, and by anonymous FTP at Columbia University. Kermit is provided free of charge for your use.

MENUS:

When you access a menu, KBCom continues to buffer RS-232 input into its internal buffer. This is especially useful if you wish to capture the contents of the screen while it is scrolling. The screen will not be updated until you exit the menus. This also allows you to exit KBCom even if the computer you are connected to is going crazy.

While working on and testing KBCom, I found that OS-9's windowing system can sometimes breaks so that it cannot open an overlay. KBCom will also be unable to open up an overlay if OS-9 does not have enough memory available to store the screen beneath the overlay. When this happens, KBCom will sound three bells (beep-beep-beep) to inform you that it cannot create the overlay window. If you are trying to exit, try again. KBCom will exit on the second attempt even if it cannot create the overlay window.

The main menu is called by pressing ALT-SPACE. You always have access to all the menu and hot-key functions of KBCom from this menu. A reverse-video bar highlights the currently selected menu option. Use the arrows to move to the desired option and press ENTER to select it. Also, pressing the character surrounded by brackets will quickly select that option. For example, in the main menu, you can press 'Q' to <Q>uit. From all menus, pressing 'X' (e<X>it) will exit that menu.

Pop-up windows requesting a single character input default to the option with the asterisk (*) next to it. The case of the input does not matter. For example, when you press 'Q' to quit, the prompt "Are you sure? (Y/N*): " is displayed. Pressing any key other than Y returns you to KBCom.

Editing is available for any prompt requesting text input. The editing keys are described in the section of the manual on conference mode. Any menu requesting a string can be aborted by pressing ENTER without entering any text. For example, if you press ALT-D to change the directory and then change your mind, pressing ENTER by itself will abort the "Set Default Directory" command. The only exception to this rule is when entering a macro or extension, where pressing ENTER without entering any text causes the key to become undefined.

In a menu where you are changing parameter values, (the terminal parameters menu, for example), use the right and left arrows to select a value for a given option. The parameter values will not become valid until you leave that menu. The right and left arrows will cycle through all of the available values. For a numeric parameter (like screen colors), you can just enter the value. For example, if you want to change the Foreground color, you can press the ALT-P to get to the parameters menu, then press 'W' to select the window parameters menu. Next, press 'F' to select "<F>oreground." Type 55 to select palette 55 for the foreground, and then 'X' to exit the terminal parameters menu and 'X' to exit the parameters menu.

PERMANENT ALT-KEYS:

The following ALT-keys are permanently defined and cannot be redefined:

ALT-SPACE	Main menu
ALT-@	Clear the screen and reset the font to normal text
ALT-/	HELP! Remember this as ALT-?
ALT-:	Send the answerback macro

MAIN MENU OPTIONS:

The following keys can be used to access the options displayed in the main menu:

ALT-A	Set an alarm.
ALT-C	Toggle conference mode on or off
ALT-D	Change the Default directory.
ALT-F	Fork a shell without closing the RS-232 path. The screen is reset to the colors in effect before running KBCom. Exit the shell by pressing ESC or by entering "ex".

ALT-G	Grab the screen and copy it to a file. Use this when you want a snapshot of the current screen. If you are in conference mode, the contents of the conference area are not saved to the file -- only the upper part of the screen is saved.
ALT-I	Show the Information Screen, which displays useful information such as the current time and session length, the alarm, the current parameter file, logfile, and VT100 LED's.
ALT-L	Turn logging on or off. When turning logging on, you are prompted for a filename. A logfile is either binary or ASCII, as set in the Terminal Parameters menu. ASCII logging throws away all non-printable characters and escape sequences and only keeps newlines and tabs. Binary logging is an exact log of all RS-232 input. If disk error occurs while writing to the logfile, the logfile will be automatically closed. A pop-up window will appear to ask if you wish to continue logging, and if so, to enter a new filename.
ALT-M	Macro menu -- use this menu to view and edit hot-keys, macros, and extensions.
ALT-O	On-line Command -- Use this to execute an OS-9 command and send the command's output to the RS-232 port. Both standard error and standard output are redirected. This is a good way to do an ASCII upload: Give the on-line command, "list filename".
ALT-P	Parameters menu -- allows you to save and load parameter files, or change parameters by selecting RS-232, window, or terminal parameter menus.
ALT-Q	Quit.
ALT-S	Put KBCom to sleep and free up the RS-232 device. Sleep until the next keypress. Use this if you want to run a terminal program in another window without quitting KBCom, for example.
ALT-Z	Zero the session timer. (Reset the clock to 00:00.)

SESSION TIMER:

The session timer allows you to keep track of how long you have been logged into a system. It is automatically set to zero when KBCom starts. You can view the session timer by calling up the information screen. For example, if I run KBCom and immediately log into a system at 6:07pm, at 7:19pm the session timer will show that I have been logged in for 1:12, or one hour and twelve minutes.

The session timer can be set to zero by pressing ALT-Z or by the \z macro sequence.

ALARM:

You can use KBCom's alarm to ensure that you only stay on Delphi for one hour, remember to eat dinner, or watch your favorite show on TV. The alarm can be set from macros by the `\a<time|message>` sequence or by the ALT-A hot key, which will prompt for the time and message. Either way, you can represent the time in several different ways:

+4:43	Four hours and forty-three minutes from now
11:30pm	At precisely 11:30pm
11:30	At precisely 11:30am
23:30	At precisely 11:30pm

If the hour is less than 12 and you do not specify pm, am is assumed. The alarm is always set for the future. For example, if the current time is 11:45pm and you set an alarm for 11:30pm, then the alarm is set for 11:30pm tomorrow. You must specify both hours and minutes, even if the hour is 0. See the example below.

If you enter no message, the alarm will go off without any display. If you supply a message, a pop-up window will display the message when the alarm sounds. You can specify an alarm without a message in a macro by leaving off the optional macro parameter. For example, the following macros will set an alarm which will sound in 90 minutes without displaying a message:

```
ALT_W      = "\a<+0:90>"
ALT_N      = "\a<+1:30>"
```

When the alarm sounds, it beeps your monitor's speaker:

beep-beep-beep beep-beep-beep beep-beep-beep.

CONFERENCE MODE:

Many BBS's or on-line services provide a conference area in which you talk to other users. Usually, what you type isn't sent to other users until you press ENTER, which allows rudimentary editing providing you can see what you are trying to edit. However, many conference facilities do not make any attempt to separate what you are typing from the messages you are recieving. As a result, you can lose track of your thoughts or be unable to edit what you are writing. Also, many systems restrict line editing to deleting characters and then retyping the line. KBCom's conference mode allows you to edit your messages locally. Nothing is sent to the host until you press ENTER. It also provides more flexible editing than many conference areas allow.

When conference mode is turned on, KBCom divides the window into two parts. Everything you type is displayed in a window at the bottom of the screen while other messages appear in the top part of the screen. This keeps what you are typing separate from other people's messages. The lower window starts out one line high, but allows you to enter 512 characters (or six 80-character lines plus part of the next). The window expands as necessary. When you press ENTER, all text up to (and not including) the character under the cursor is sent, and the conference window returns to one line.

The following keys perform special functions while in conference mode:

Right/Left arrow	- Move one character right or left
SHIFT-Right/Left arrow	- Move to the end/beginning of text
CTRL-Right/Left arrow	- Insert a space/delete the current character
Up/Down arrow	- Move up or down one line if possible
BREAK	- Discard current text and start over

Those of you who have used Kevin Darling's SCF editor will notice that most of the editing keys are the same. Also, notice that you lose the BREAK macro while in conference mode.

CONF_NOBUFF and control characters:

Conference mode normally discards all control characters except BEL (Control-G) and beeps to warn you that the character you pressed is illegal. No other control codes can be entered into conference mode; however, you are able to send some control codes immediately while in conference mode (without disturbing the contents of the message you are editing). The parameter CONF_NOBUFF, which can only be set in a parameter file, defines which control characters are sent as they are typed (and therefore not buffered). If you want to send ^S and ^Q while within conference mode, include the following line in your parameter file:

```
CONF_NOBUFF = "SQ"
```

Note that the letters S and Q are not prefaced by the caret '^'.

PARAMETERS MENU:

The parameters menu allows you to call up the window parameters menu to set window colors and font parameters, the RS-232 parameters menu to set the RS-232 parameters, or the terminal parameters menu to set KBCom's parameters. This menu also contains the options to load and save parameter files. The option resave parameters saves the current parameters to the last referenced parameter file. (The last parameter file which load loaded or saved.) Finally, the option load without autoexec will load a parameter file without executing the AUTOEXEC macro.

All parameters can be modified within KBCom with the exceptions of DEFAULT_DIR and CONF_NOBUFF. DEFAULT_DIR is the default directory that KBCom changes to after loading the parameter file. The only way you can set this parameter is by editing the parameter file. However the current value of DEFAULT_DIR (as specified in the most recently loaded parameter file) is preserved and saved each time you save a parameter file.

Keep in mind the vast differences between complete and relative paths when you define DEFAULT_DIR. A complete pathlist specifies the entire path to the directory, including the drive name, and always begins with a slash. For example, /DD/DOWNLOADS/APPLICATIONS is a directory two levels deep on the disk in drive /DD. You are always guaranteed to arrive in the specified directory when using a complete pathlist -- provided the directory exists. You will probably want DEFAULT_DIR to be a complete path.

A relative pathlist, as the name implies, is relative from the current directory. For example, assume we are in the /DD/DOWNLOADS directory and we type:

```
chd APPLICATIONS
```

OS-9 assumes you want the /DD/DOWNLOADS/APPLICATIONS directory. But if you are in the /DD/USERS directory when you enter that command, OS-9 will assume you mean the directory /DD/USERS/APPLICATIONS.

PARAMETER FILES:

Parameter files are ASCII files with an extension of `.kbc` and can be edited by any text editor. If you run KBCom without giving the name of a parameter file on the command line, KBCom uses `default.kbc`. If you provide a parameter file without a full path to it, the parameter file directory is assumed. Finally, the extension of `.kbc` is assumed and added to the filename you enter if necessary. Before a parameter file is loaded, all parameters are reset to their default values; any parameters not specified in a parameter file are guaranteed to be reset to their defaults. The same is true for macros, hot-keys, and extensions. The default values for RS-232 parameters are the settings KBCom started with (the xmode values for that device) and full duplex. Parameters can appear in a parameter file in any order. The tab settings are always saved.

Enter integer and character parameters in a parameter file as follows:

```
FORE    = 33
TAB     = 5, 8, 16, 24
ALT_F   = S
```


String values in a parameter file are surrounded by double quotes ("). You can enter control characters in a string by preceeding the character with a caret (^). To enter control-M (carriage return) in a string, use the two characters ^M. Characters 27-31 are represented by ^[, ^|, ^], ^^, and ^_, respectively. You can enter a DEL (127, or delete) by using ^?. Alt-keys are entered into a string by preceeding the character with a tilde (~). To insert ALT-L into a string, use the two characters ~L. To enter a caret (^), tilde (~), or double quote (") into a string, "quote" it by preceeding it with a backslash (\). When the parameter file is read in, these backslashes will be removed from the string. Backslashes appearing before characters other than these three are left in the string. For example, you could define the macro:

```
ALT_A = "This string sends a caret \^ " -
        "tilde \~ and quote \" ^M"
```

A parameter file allows for continuation lines for very long macros and extensions by continuing onto the next line. To indicate a continuation line, insert a hyphen (-) at the end of the line. To continue a string across to the next line, close the quotes, add a hyphen, and open the string with a quotation mark on the next line. For example:

```
ALT_Z   = "This could be a string that is " -
          "quite long and is " -
          "thus continued across four" -
          "lines in the parameter file!!!"
```

INCLUDE FILES:

You can use parameter include files to define parameters that would otherwise be defined in many parameter files. For example, let's say that file_transfers.kbc in the parameter file directory contains a set of extensions which invoke file transfer programs. Any parameter file which contains the line:

```
INCLUDE = "file_transfer.kbc"
```

will load the contents of `file_transfers.kbc`. If you change the value of any parameter in this file, that parameter's value is changed for all parameter files that include this file. This can save you a lot of typing! INCLUDE files follow all rules applying to parameter file filenames. You may redefine any parameter which occurs in an INCLUDE file by defining it again after the INCLUDE statement. Currently, a parameter file may include only one file; an include file may not include another file.

When saving parameters, any hot-keys, macros, or extensions which were defined in an INCLUDE file will not be written to the parameter file. Instead, the

INCLUDE declaration will be written before all other hot-keys, macros, and extensions. KBCom only keeps track of where key definitions are defined; all other definitions will always be written to the parameter file.

WINDOW PARAMETERS:

The window parameters menu lets you set the screen colors that KBCom will use. Foreground, Background, and Border set the foreground, background, and border colors. A foreground color is the color of the text while a background color is the color of the region underneath the text. KBCom emulates BOLD text on a text screen by using different palettes as defined by Bold Foreground and Bold Background. Menu Foreground and Menu Background set the colors of the menu screens. The Shadow parameter is the palette number for the overlay window shadows. Shadows aren't used if you run KBCom on a two-color window; instead, the menus are in inverse video. Finally, USE_FONTS is either ON or OFF to indicate whether or not you are using the VT100 special fonts. KBCom must be run on a graphics screen to use the fonts.

Note that on a four-color graphics screen, only the Foreground, Background, Menu Foreground and Menu Background color options are used, while on a two-color graphics screen, only the Foreground and Background color options are used.

In a parameter file, the window parameters are saved as:

Parameter	Default	Value Range	Meaning
FORE	63 (white)	0-63	Foreground palette
BACK	9 (blue)	0-63	Background palette
BORDER	9 (blue)	0-63	Border palette
MENUFORE	0 (black)	0-63	Menu foreground palette
MENUBACK	62 (yellow)	0-63	Menu background palette
BOLDFORE	54 (orange)	0-63	BOLD foreground palette
BOLDBACK	9 (blue)	0-63	BOLD background palette
SHAD_COLOR	0 (black)	0-63	Menu shadow palette
USE_FONTS	1 (OFF)	0-1	Special Fonts ON (0)/OFF (1)

RS-232 PARAMETERS:

KBCom initializes the RS-232 parameters from the current settings in the device it is run on. You can examine and change these settings with the xmode command before running KBCom. Settings in a parameter file override the defaults. The RS-232 parameters include:

Parameter	Default	Value Range	Meaning
BAUD	See Above	0-7	Baud rate; 110 to 19,200 bps
PARITY	See Above	0-4	Parity
STOPBITS	See Above	1,2	Number of Stop Bits
DATABITS	See Above	7,8	Number of Data Bits
DUPLEX	0 (Full)	0-1	Full Duplex (0)/ Half Duplex (1)

The Baud rate values (0-7) are the same as those used with the xmode command:

Baud Rate	Parameter Value	Baud Rate	Parameter Value
110	0	2400	4
300	1	4800	5
600	2	9600	6
1200	3	19,200	7

The Parity values are as follows:

Parity	Parameter Value
None	0
Odd	1
Even	2
Mark	3
Space	4

Some computer systems are picky about RS-232 settings, it may take some work to connect to a computer system for the first time. Nearly all hosts use one stop bit. Space and Mark parities are rarely used. I have found that I can connect to almost all systems using 8 data bits and no parity. When using no parity, you should use 8 data bits; however, when using any other parity, you should select 7 data bits. (The 8th bit is the parity bit.)

Use half duplex for a host that will not echo your keypresses. If you have DUPLEX set to FULL and see nothing on your screen when you are typing, try setting DUPLEX to HALF. On the other hand, if you see everything you type doubled, then make sure that DUPLEX is set to FULL.

TERMINAL PARAMETERS:

The terminal parameters are defined below:

Arrow Keys:	OS-9:	Use OS-9 standard arrow keys.
	OS-9/Del:	As above, but the left arrow sends DEL.
	Emulation:	Send arrow sequences required by the emulation.
Backspace:	Normal (nondestructive) or Destructive. A destructive backspace erases characters as the cursor backs over them.	
	Emulation:	Select the emulation: VT100, VT52, ANSI, OS-9, or CRT.
Newline Mode:	ON or OFF, the equivalent of VT100 Newline Mode. If ON, the ENTER key generates two characters: carriage-return plus line-feed. Otherwise, it generates a carriage-return only.	
Cursor:	Flashing, Steady (ON), or Off.	
7-bit Mask:	ON or OFF. If ON, the 7th bit of each incoming character is stripped (forcing the character into the range 0-127). Otherwise, the 7th bit is preserved.	
VT Keypad:	ON or OFF. If OFF, the ALT-0 through ALT-9 keys can be used as macros, hot-keys, and extensions. Otherwise they send Application keypad codes.	
Logfile Type:	ASCII or Binary. An ASCII logfile contains text only. Control characters and escape sequences are ignored. A binary logfile is an exact copy of all RS232 input (after the 7th bit has been masked if that feature is enabled).	

The terminal parameters are saved in parameter files as follows:

Parameter	Default Value	Range	Meaning
ARROW_KEYS	0 (VT100)	0-2	Emulation, OS-9, OS-9/Del
BACKSPACE	0 (Normal)	0-1	Normal, Destructive
EMULATION	0 (VT100)	0-4	VT100, VT52, ANSI, OS-9, or CRT
NEWLINE	1 (OFF)	0-1	On or Off
BLINK_CURSOR	0 (Steady)	0-1	Steady Cursor, Flashing Cursor, or None
7BIT_MASK	0 (ON)	0-1	On or Off
VT_KEYPAD	0 (ON)	0-1	On or Off
LOGFILE_TYPE	0 (ASCII)	0-1	ASCII or Binary

FLAGS:

KBCom contains a number of flags which cannot be directly set. All of these flags are related to VT52, VT100, and ANSI emulation and can be set or reset by the remote system. Flags can only be TRUE (ON) or FALSE (OFF). The CONF parameter is included in this section because it occurs in a parameter file in the same manner.

The flags are:

Parameter	Default	Meaning
CURSOR_MODE	FALSE	Application (TRUE) or Cursor (FALSE) arrow keys
ORIGIN_MODE	FALSE	Absolute (TRUE) or Relative (FALSE) origin
WRAPAROUND	TRUE	On or Off
APPLICATION	TRUE	Application keypad (ON) or Numeric keypad (OFF)
UNSOLICITED	FALSE	Allowed (TRUE) or Not allowed (FALSE)
CONF	FALSE	Conference mode ON/OFF

APPLICATION and CURSOR_MODE change the action of the application keypad (if VT_Keypad is enabled) and the arrow keys (if ARROW_KEYS is set to Emulation), respectively. ORIGIN_MODE is related to the VT100 scrolling region. If WRAPAROUND is TRUE, then when you enter a character into the last column of a line, the screen will scroll and the cursor will move down to the next line. Otherwise, the cursor will remain in the last position of the line. If UNSOLICITED is TRUE, KBCom will send a report to the remote system any time RS-232 parameters are modified. I have not yet encountered a system which responds to such reports, however!

You will probably never have to change any of these flags. Most systems which expect VT52, VT100 or ANSI emulation will set these flags to the desired values.

In a parameter file, FALSE is indicated by 0 and TRUE by -1. For example, you would turn conference mode on by including the following line in a parameter file:

```
CONF = -1
```

HOT-KEYS, MACROS, AND EXTENSIONS:

KBCom provides 38 redefinable keys, including ALT-A through ALT-Z, ALT-0 through ALT-9, ALT-:, and BREAK. ALT-: and BREAK can be defined as macros only, but the rest can be defined as hot-keys, macros, or extensions. A key that is not defined causes no action.

A hot-key is a key that calls a menu or a pop-up window. The main menu contains most of the possible hot-key values; the hot-key to invoke a menu item is the character between braces. If you define ALT-E to send the hot key 'Q', then pressing Alt-E will call up the <Q>uit pop-up window. Hot keys are useful if you

define a macro or extension over the original key for that menu item. For example, if you define an extension on ALT-D to dial a computer, you can define ALT-W as the hot-key 'D' to change the working directory. In addition to the choices listed in the main menu, you can also define a hot key to 'W', 'R', or 'T' to call up the Window Parameters, RS-232 Parameters, or Terminal Parameters menu directly.

Macros allow you to save key-strokes by storing commonly used strings that you can send all at once. As a simple example, if you define the ALT-H as the string "`/send myfriend Hello there! How are you?^M`", every time you press ALT-H, KBCom will send that string just as if you had typed it. Notice that a carriage return is sent at the end of the macro by the ^M (control-M). Special character sequences to place control characters in macros are described below in the section on special characters. Macros also allow more powerful features such as the ability to pause and wait for a string to be recieved before continuing and the ability to prompt the user for a string to include as part of the macro. These features are described in detail in the sections on special features and subsitutions below. For example, if you define ALT-H to be the macro:

```
/send %r<Hi To: > Hello there! How are you?^M
```

each time you invoke that macro, a pop-up window will appear, asking: "`Hi To:`". The string you enter will replace "`%r<Hi To: >`" in the macro definition.

A macro can contain a maximum of 512 characters. Also, the cursor is turned off while a macro is executing to show that the macro is controlling what is being sent to the modem. Once the macro finishes, the cursor returns to the state it is set to in the Terminal Parameters menu. You can abort a macro by pressing any key while it is executing.

Macros have full access to KBCom's menus and indeed may invoke other macros. This ability is fully described in section about special characters. The only restrictions on macros calling macros are that a macro should not invoke itself, and that the combination of the chained macros (at the moment of chaining) must be shorter than 512 characters. Characters which have already been sent through the modem do not count toward this limit when a macro invokes another macro.

There are two special macros: AUTOSTART and ANSWERBACK. The AUTOSTART macro is executed automatically upon loading the parameter file that defines it, right after the file has finished loading. The ANSWERBACK macro is sent when you press ALT-: or when the remote computer asking for it.

Extensions are essentially other OS-9 programs that you call from within KBCom. You would use an extension to run a standalone file transfer protocal such as Kermit, XModem, or YModem, for example. If you have ALT-F defined as the extension "free /D0", any time you press ALT-F, KBCom will run that command to

show you how much free space is on drive /D0. Extensions are described in great detail in the creating an extension section. Extensions are allowed to be 256 characters long.

Most ALT-keys are undefined by default. The only keys that default to a value are those that correspond to entries in the Main Menu. For example, ALT-Q defaults to the hot key 'Q', while ALT-Y is undefined by default.

MACRO EDIT MENU:

Use the macro edit menu to view the current value assigned to an ALT-key. You can select a key to view by using the arrow keys, or you may press the key corresponding to the macro you wish to view. (Remember, ALT-':' sends the answerback macro, so you may press '`:`' to edit the answerback macro.) As in all other menus, pressing 'X' exits the menu.

The value of the selected key is displayed in the window at the bottom of the Macro Edit menu. For a long macro or extension, only the first part of the macro string or extension command line is shown. Press ENTER to edit the selected key's value; this displays an overlay giving you the option of exiting or setting the value of the key as a hot-key, macro, or extension. If you edit a macro or extension that is already defined, you will be able to edit the old string. The editing available for entering a macro or extension is the same as the conference mode editing.

CREATING A HOT-KEY

To define an ALT-key as a hot-key, all you need to do is press the desired hot-key. For example, if you selected ALT-W, pressed ENTER, and then H to define that key as a hot key, all you need to do is press D to define ALT-W as the hot-key D. The available hot keys are those defined in the Main Menu plus W, R, and T to call up the Window Parameters, RS-232 Parameters, and Terminal Parameters menus, respectively.

SPECIAL CHARACTERS:

When entering macros and extensions, you can enter a control code by prefacing it with a caret (^). This works when you are entering it in a KBCom prompt as well as in a parameter file. For example, ^A (caret, A) in a macro is treated as CTRL-A. Non-alphabetic control codes can be entered as follows:

27 (Escape)	^[	28	^|
29	^]	30	^^
31	^_	127 (Del)	^?

You cannot enter a NULL in a macro or extension.

You can also enter ALT-keys in macros and extensions. To do this, preface the ALT-character with a tilde (~). Do NOT enter the ALT-character directly! Thus, ~P (tilde, P) in a macro is treated as ALT-P.

CREATING A MACRO:

Creating a macro involves typing it in, either in a parameter file or in the macro edit menu.

Any ALT-key in a macro acts as if you typed it yourself. Using the above example, ~P acts just as if you pressed ALT-P while the macro executed. This gives macros full access to the menus and allows macros to call other macros. For example, you could have a macro defined as:

```
~L/r0/delphi.log^M
```

if you want to turn logging on in a macro. Notice in the example above that you must end any text response to a menu with ^M (ENTER), just as you would press ENTER after typing the string "`delphi.log`" if you were turning on logging yourself.

To give macros better access to the menus, ^S has a special meaning in menus when a macro is executing. As an example, ^S03 means "select menu option 3." Two digits are expected to follow the ^S. To set the baud rate, you could define a macro containing: "`~PRB^S03XX`" -- this macro enters the parameter menu (~P), enters the RS-232 menu (R), selects baud rate (B), then selects baud rate 3 (^S03), which is 1200 bps, and then exits both menus (XX).

CREATING AN EXTENSION:

When creating an extension, you are asked several questions. First, enter the extension, which can be any command that you can enter at a shell prompt. You may take advantage of all features of the shell, including wildcards (if you are using a shell that supports wildcards), pipes, and redirection.

You are then prompted with the question, "Preload?" Do you want KBCom to preload the program? That is, do you want to load the program into memory in advance? You can preload frequently used extensions so they don't have to be loaded from disk each time you use it, which saves time. KBCom will unlink each extension when you undefine or redefine that extension, or quit KBCom. You can provide a list of modules to be preloaded, separated by spaces. KBCom doesn't check if the modules are already in memory, so if you specify the same module for more than one extension, it will be preloaded once for each occurance!

Next, you are asked "Pause for a keypress when done?" If you answer 'Y' to this question, KBCom will wait for a keypress when the extension is finished. This is useful for some programs, especially if you run them on an overlay window and you want to see the output.

The next prompt is "Close/Pass/Ignore RS-232" If you answer 'C' then KBCom will close its path to the RS-232 device before calling the extension and reopen it when the extension has finished. If you answer 'P,' KBCom will pass its paths to the extension. The paths include the RS-232 device on path 0, the window on path 1, and the logfile (or window if no logfile is open) on path 2. Answing 'I' will cause KBCom to ignore RS-232 input while the extension runs without actually closing the path to the RS-232 device. If you answer 'N', KBCom will do none of the above, so paths 0, 1, and 2 will contain their normal values (all will be paths to the window).

The next question is "Open an overlay before calling?" This means, do you want an overlay window opened to run this extension in? If so, you are prompted for its starting row and column position and its width and height.

Finally, the last question is "Number of Beeps when done (3/2/1/0*): " which asks you how many times you want KBCom to beep to signal that the extension has finished. This allows you to go into another room while waiting for a long download to finish, for example.

Once you have entered the text of the command and answered the above questions, you have created an extension! Since creating an extension can be complicated, several examples are provided in the sample parameter files.

SUBSITUTIONS:

Macros and extensions allow subsitutions. A subsitutions begins with a percent sign (%) in the text of the macro or extension; the character following the percent sign defines which subsitution follows. When invoked, the subsitution is replaced by its value. The valid subsitutions are:

%%	% (allows you to use a percent sign)
%b	The baud rate (110, 300, 600, etc...)
%c<prompt|which>	Get one character out of the set 'which'
%d	The number of data bits (7 or 8)
%n<prompt>	Get a secret string
%p	The parity (N, E, O, M, S)
%r<prompt>	Get a string
%s	The number of stop bits (1 or 2)
%t	The name of the RS-232 device

Use one of the three prompted subsitutions (%c, %n, %r) when you need user input before sending the macro or executing the extension. A pop-up window will be displayed (with the given prompt) asking for input. If the prompt string is null, the prompt will be the extension or macro up to that point. The %r subsitution is useful for passing filenames to an extension, and the %n subsitution is useful for prompting for a password in a macro. The %c subsitution returns one character out of the list 'which,' defaulting to the last character in that string if a character not in that string is pressed. The characters in 'which' are added to the prompt to show which characters are allowed. If '`|which`' is left out of the %c subsitution, any character is accepted. Look at the example parameter files to see how these work.

If you press BREAK in response to the %c subsitution, then the macro or extension is aborted. If you press BREAK when entering text for the %r or %n subsitutions, one of two actions takes place. If there is text in the pop-up window, then that text is erased just as in conference mode. The macro or extension will be aborted, however, if you press BREAK when there is no text in the pop-up window.

SPECIAL ACTIONS:

Macros allow special actions, which are signaled by a backslash:

\\	--	Send a backslash '\'
\?x	--	Wait for the character following '?' (x, in this case)
\=	--	Pause for two seconds
\a<time|Message>	--	Set the alarm to sound at the specified time and display the given message on a pop-up window. The message may contain any character except '>' and '|'
\p<n>	--	Pause for n seconds
\r<String>	--	Recieve (wait for) 'String' from the modem before continuing. The string may contain any character except '>' and '|'
\z	--	Zero the Session Timer

The alarm and session timer are described in the section on alarms.

KEY DEFINITIONS:

PARAMETER FILE:

Hot keys are stored in a parameter file as:

```
ALT_W = D
```


Notice that there are no quotes around the D. The example above defines ALT- as a hot key which changes the current working directory.

Macros are stored in a parameter file as:

```
ALT_E = "This is a macro!^M"
```

Extensions are stored in a parameter file by listing the command string, the option (stored in a string), the names of any preloaded files (stored in a string), and finall the row, column, width, and height of the overlay to run the extension in. Al arguments are separated by commas. The list of preloaded files and overlay windo size and position are omitted if not used. The options string contains, in order, 'Y or 'N' to indicate if it waits for a keypress after the extension is completed; 'C', 'P' or 'N' to indicate if the RS-232 path is Closed, Passed, or Neither; 'Y' or 'N' to sho if an overlay is opened for this extension, and a digit 0-3 indicating the number o times KBCom beeps when the extension is finished. For example:

```
ALT_T= "Transfer -line %t -baud %b %c<" -
       "Upload/Download|ud> %r<File:>", -
       "NCN3", "transfer"
ALT_K= "kermit -log /r0/kermit.log"  -
       "%c<Command:|srcf> %r<File: >","NPY3",
       5,7, 70,10
ALT_9= "%r<OS-9 Command: >", "YNY0", 5,7, 70,1
```

For ALT-T, the program "`transfer`" is preloaded. ALT-K and ALT-9 bot open an overlay for the extensions to be run in. When the ALT-T extension ha completed, KBCom beeps three times to let you know it has finished. KBCom wait for a keypress before continuing after ALT-9 has finished. Notice that ALT-9 allow you to run any OS-9 command without flipping windows. You can find out th current directory, for example, by pressing ALT-9 and typing "`pwd`". If you have dire need to format a disk and have no windows free, you can press ALT-9 and ente the command to format a disk.

KNOWN BUGS:

VT52, VT100, ANSI emulation:

VT52, VT100, and ANSI terminals do not always scroll when a character is place on the last positiion of a line; however, OS-9 windows always scroll when a characte is placed in the last position of a line. Thus, to prevent the screen from scrolling whe it shouldn't, any character placed in the bottom right corner of the screen will not b displayed until the screen scrolls.

ALARM AND CLOCK:

If you are using the Tandy floppy disk controller or any other halting floppy disk controller, and you don't have a real-time clock, then the system clock will slowly lose time if you access the floppy disks. This will cause any alarms to go off later than expected, and the session timer will indicate less time than you've actually spent on the system. The actual amount of lost time will depend on how much you access the floppy drives.

Also, if you set the system clock while KBCom is running, the session timer and alarm will be incorrect until you zero the session timer and re-set the alarm.

APPENDIX A -- EXAMPLE PARAMETER FILE

Two sample parameter files are listed below and explained separately:

```
---------- cocofest.kbc ----------
APPLICATION = 0
BAUD = 3
ARROW_KEYS = 1
BACKSPACE = 1
BLINK_CURSOR = 1
TABS = 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113, 121, 129
INCLUDE = "file_transfers.kbc"
ALT_A ="atdt*70,4513362^M\r<CONNECT 1200^M>\=a\r<in:>delphi^M"
        "\r<Username:>yourusername^M%n<Password: >^M\z"
ALT_Z ="\a<7:00pm|Watch the Simpsons!>"
CONF_NOBUFF = "XZOSQ"
DEFAULT_DIR = "/dd/downloads"
---------------------------------------------------------------------------
```

This parameter file will dial into Delphi using Tymnet. The lines of this paramete file: turn off the application keypad, set the baud rate to 3 (1200 baud), set arrow key to OS-9 arrows, set the backspace to the destructive backspace, set the cursor so i blinks, and set the tabs. Next, it includes the parameter include file file_transfers.kbc which is below and contains several extensions set up to transfer files using kermi and XYModem.

The macro ALT-A dials into Delphi using Tymnet. The Tymnet phone number use in this example is for Atlanta, Georgia. This macro first calls Tymnet, turning off cal waiting with the "*70,". The comma instructs a Hayes compatible modem to wait fo a couple seconds before continuing. The macro next waits for the string "CON NECT 1200^M" which will be recieved when the modem has connected to Tymnet After pausing for two seconds with the "\=", KBCom sends a lowercase a, and the waits to receive the string "in:" from the Tymnet prompt "please log in:". Afte receiving this prompt, KBCom responds with "delphi^M", waits for Delphi' "Username:" prompt, and sends your username and password. Finally, the sessio timer is zeroed so you can keep track of how much time you spend online.

The macro ALT-Z sets an alarm which will go off at 7:00pm with the message "Watc the Simpsons!" The control characters CTRL-X, CTRL-Z, CTRL-O, CTRL-S, an CTRL-Q are allowed to be entered while in conference mode with the CONF_NOBUF line. Finally, KBCom sets its default directory to the directory /DD/DOWNLOADS

```
---------- file_transfers.kbc ----------
ALT_B =  "xymodem -ybg%c<Upload/Download|ud> -log /r0/xy.log" -
         "%r<File: >", "N" "PN3"
ALT_K =  ":kermit -%c<Command>%r<Options: >l %t %r<Files: >", -
         "NNY3",  0,5, 80,10
ALT_W = "%r<OS-9 Command: >", "YNY0", 5,7, 70,10
ALT_X = "xymodem -xc%c<Upload/Download|ud> -baud %b -log" -
         "/r0/xy.log %r<File" ": >", "NPN3"
ALT_Y = "xymodem -y%c<Upload/Download|ud> -baud %b -log -
         "/r0/xy.log %r<File:" " >", "NPN3"
---------------------------------------------------------------------------
```

This parameter file contains five extensions that are commonly used in all other parameter files. The first, ALT-B, runs XYModem using the YModem batch protocal to transfer files, prompting for the direction of file transfer and filename(s). ALT-X and ALT-Y also run XYModem, but as XModem-CRC and YModem (nonbatch) transfers. ALT-K runs kermit, prompting for the kermit command, options, and files. Kermit is run in an overlay window. ALT-W allows you to execute any OS-9 command in a pop-up window; it waits for a keypress when the OS-9 command has finished so you can see the output.

APPENDIX B -- XYModem

NAME: XYModem 1.0, © Copyright 1990, Edward W. Kuns

USAGE: xymodem {-command} {-flags} {file ....}

If you run XYModem with not parameters, it will display a help screen and exit. The case of commands and flags doesn't matter. Place the parameters of flags which have parameters following the parameter (with an optional space between them). You must use a dash (-) to begin a command or flag, but additional flags may be run together. Several examples are given at the end of this document.

PURPOSE:

XYModem is a file transfer program which allows you to send and receive files using XModem, YModem, and YModem batch protocals, and was designed to work well with KBCom. Although the version of kermit that is distributed with KBCom is public domain, XYModem is NOT.

COMMANDS:

XYModem needs a direction for the transfer as well as a file transfer type. Not specifying a direction is an error and will cause XYModem to quit; however, if you do not specify a transfer type, XYModem will assume you want YModem (non-batch) using CRC block checks. The commands are listed below:

-U or -S	Upload file(s)
-D or -R	Download file(s)
-X	Use XModem with checksums
-XC	Use XModem with CRC's
-Y	Use XModem-1k (YModem) with CRC's
-YB	Use YModem batch with CRC's

FLAGS:

Q	Quiet mode, don't display a status window
V	Verbose level (from 0-5 'V's)
A	The following files are ASCII
B	The following files are binary
G	Guess the filetype of following files
L	Use the specified RS-232 device (line)
BAUD	Set the baud rate
STOP	Set the number of stop bits
O	Overwrite file if it already exists when downloading
E	Erase partial download if a download fails or is cancelled
LOG	Keep a logfile

RS-232 DEVICE:

If you use the L flag, XYModem will open up a path to the RS-232 device. Otherwise, XYModem assumes that path 0 is connected to the RS-232 device, path 1 is connected to the window, and path 2 is connected to a logfile if different from path 1. If paths 0 and 1 are the same, then XYModem is automatically put into QUIET mode.

STATUS WINDOW:

When XYModem is running, it normally displays a status window. The status window displays the type of transfer and a running tab of many useful values, such as the number of blocks and bytes transferred, the error count, and the file being transferred. The bottom of the status window is a 4-line message window for messages. The higher VERBOSE is, the more messages you will see here! The message window will scroll when necessary.

FILENAMES:

When sending a file in YModem batch, filenames are sent without any modification. ONLY the filename is sent, however not the path to that file. For example, if you upload the file "`/dd/cmds/icons/tree.icon,`" the filename is sent as "`tree.icon.`" When a filename is received, it is converted, if necessary, to be a legal OS-9 filename. If the first character of the filename is not a letter, then "`x_`" is added to the beginning of the filename. Any illegal characters are converted to underscores '_'. Finally, the filename will be truncated to 29 characters.

If you are downloading, and the file being downloaded has the same name as a file that already exists, XYModem will try to avoid the filename collision by adding `.0`, `.1`, etc to the filename until it finds a filename that doesn't exist. If you specify the O flag, existing files will be overwritten.

You don't need to specify filenames when downloading in YModem batch. If you do, then the filenames you specify on the command line override the incoming batch filenames. This is useful when the YModem Batch filenames are long or gibberish.

FILE TYPES:

When transferring an ASCII file, XYModem performs a number of translations on the data. All non-ASCII characters are discarded, and the end-of-line character is converted as necessary. No conversions are performed on binary files. GUESS mode decides a file is binary if the first packet:

1) Contains any characters with the 8th bit set (128-255)
2) Contains any control character other than Carriage Return, Linefeed, Tab, or XModem padding characters (NULL or control-Z).
3) Contains a NULL followed by a non-NULL
4) Begins with a NULL

So far, XYModem has always chosen the correct file type; however, it is possible that there are unusual files that will fail this test. For such files, you will want to explicitly declare the file type. Since the test is performed only on the first packet, GUESS mode is more useful for YModem or YModem batch than it is for XModem.

YModem batch does not send the size of a file which is being sent in ASCII mode, because the size is not known. The ASCII translations may (and probably will) change the file size. YModem batch ignores the file size of an incoming ASCII file.

The A, B, and G flags stay in effect until changed. See the examples below.

EXAMPLES:

To upload using YModem Batch, f.txt as ASCII and `bin_data` as binary, you could use the command line:

```
xymodem -ybu -baud 1200 -log /r0/xy.log -a f.txt
-b bin_data
```

XYModem also uses the current paths; it does not open a path to the RS-232 device, since it was not given the `l` option. Information is logged into the file, `/r0/xy.log`, and the transfer takes place at 1200 bps.

```
xymodem -ybdgoe -l /t2 fantastic.gif onetwo.icon
```

will download using YModem batch. Files recieved that already exist will be overwritten. Partial downloads will be erased. The first two downloaded files will be named fantastic.gif and onetwo.icon. Any additional downloaded files will keep their YModem batch names. The filetypes will be guessed based on the contents of the first packet of each file.

APPENDIX C - Kermit

NAME: OS-9 Kermit Version 21 Release 5

AUTHORS:

A great many people have worked on this kermit over the years. The last two authors, who have made significant changes and additions (and are the only authors after 1985!), are James Jones and more recently, Simmule Turner.

USAGE: kermit {-command{flags}} {line} {char} {file ...}

Commands:

s	Send file(s) (use '-s -' to send from stdin)
r	Receive
k	Receive to stdout
g	Get remote file(s) from server
x	Enter server mode
f	Finish remote server
c	Enter connect mode

Flags:

l	Set communication line (eg, /t2)
e	Set the escape character for connect mode
i	Binary transfers or OS-9 to OS-9 (image mode)
d	Increase debug level
X	Disable XON/XOFF
F	No filename case conversions
1	Select the one-byte checksum block check
2	Select the two-byte CRC block check, which is more accurate
8	Use 8th bit quoting for a 7-bit data path

DESCRIPTION:

Kermit provides reliable file transfer and also acts as a primitive terminal program. It has been implemented on many different computers, including microprocessors. Kermit can transfer ASCII as well as binary files. Kermit's commands and flags are similiar to those used in C-Kermit, a popular version of kermit running on many UNIX machines.

The arguments to kermit are a set of flags with no spaces between them. Notice that the case of flags and commands is important! The flags can occur in any order.

There must be one and exactly one command. If you run kermit without any commands or flags, kermit will print out a 'help' message. Place the RS-232 device and escape character after all flags in the order the l and e flags were placed in. The following command, for example, runs kermit on `/t3` in connect mode with the escape character '$' and without performing filename translations:

```
kermit -ceFl $ /t3
```

Kermit can transfer binary and text (ASCII) files. However, unless you use connect mode or are transferring files from OS-9 to OS-9, you cannot transfer both kinds of files in one shot; you will have to run kermit once to transfer binary file(s) and once again to transfer text file(s). Kermit can also send and recieve binary files (files using all characters 0-255) on a 7-bit data path (if you must use parity, for example). When sending text files, the end-of-line character is translated from carriage return for OS-9 to carriage return + linefeed, linefeeds are dropped when recieving. The remote kermit will translate the carriage-return + linefeed into it's end-of-line character. All possible filetypes are listed below:

(no option)	7-bit ASCII text file, end-of-line conversions
8	8-bit text file, end-of-line conversions, quote 8th bit
i	binary data
i8	binary data on a 7-bit connection, quote 8th bit

There is no way to send an 8-bit text file without quoting the 8th bit unless you send the file without end-of-line conversions.

Use the l flag to tell kermit what device to connect to. If you do not use this flag, kermit assumes that the default paths are connected to the remote system. (This is correct, for example, when kermit is being used on a BBS.) If you do not wish XON/XOFF flow control to be used during the transfer, use the X flag.

Kermit offers two types of block checks: one-byte checksums and two-byte CRC's. All implementations of kermit support the checksums; however, not all support CRC's. The CRC block check is a more robust check of the integrity of the data, although checksums are probably sufficient for most connections.

Unless you specify the F flag, kermit will change the case of filenames. When sending, filenames will be converted to uppercase, and when receiving, filenames will be converted to lowercase.

Following the command and flags, enter the filenames of the file(s) to be transferred for the send and get commands. When kermit is recieving, it will normally attempt to store the file with the same name it has on the remote host. Outgoing

filenames are converted to uppercase unless you have specified the F flag. Filenames containing a slash (/) will have everything up until and including the last slash removed before the file is stored on disk or sent to the remote kermit. (Filenames are sent as you type them for the Get command, however, and converted as above when the file(s) are actually being recieved.)

COMMANDS:

SEND: (`-s` command)

Kermit will send the files listed on the command line to the remote kermit. The files will be sent in the order they appear on the command line.

RECEIVE: (`-r` and `-k` commands)

Recieve files from the remote kermit. Kermit does not request files; the remote kermit must be in SEND mode. The incoming files will be stored with the same names as on the remote system. If you enter RECIEVE mode with the `-k` command, kermit will not open a file for the output. The output of kermit will be sent to standard output.

GET: (`-g` command)

To use this mode, the remote kermit must be in SERVER mode. Kermit will request files from the remote kermit, and assuming those files exist, will recieve them.

SERVER: (`-x` command)

Kermit will enter SERVER mode. All control is given to the remote kermit, which can make requests. Kermit will exit when given the FINISH command by the remote kermit.

FINISH: (`-f` command)

Send the FINISH command to the remote kermit server to make it exit.

CONNECT: (`-c` command)

CONNECT mode allows kermit to function as a primitive terminal program. Press the escape character, which is '#' unless you have changed it with the e flag, to access

special features. Kermit will wait for the next keypress after you press the escape character. You can send the escape character to the remote host by pressing it twice. Other functions available are:

! or p	Push to a shell
?	Help
0	Send a NULL character
8	Toggle 8-bit and 7-bit mode
c	Enter Kermit COMMAND mode, below
q	Quit logging
r	Resume logging

COMMAND MODE:

Kermit's command mode allows you to send, recieve, or get files, and offers access to many OS-9 commands and commands on the remote machine. In the list below, an asterisk in a command means that you only need to type the command up to that point to execute it; the command is compared only up tp the position of the asterisk. The list isn't quite alphabetical; it reflects the order the commands are checked in. For example, if you enter "`co`," kermit will check `copy` (but you didn't type the necessary three characters) and then `connect`, which will match. If you type "`close`," the string will not match "`close session`" because you didn't enter enough characters; but it will match "`connect!`" The available commands are:

!	Push to a shell
?	Display available commands
bye	Close the remote server
close s*ession	Close the logfile
cop*y	Copy a file (OS-9 command)
cwd	Change working directory (OS-9 chd command)
c*onnect	Return to connect mode
dat*e	Display the current date and time (OS-9 command)
dial	Dial a phone number, not currently implemented
dir	Display a directory (OS-9 command)
del*ete	Delete file(s) (OS-9 command)
get	Get a file from a remote kermit server
hel*p	Display available commands
ima*ge	Toggle image mode
log s*ession	Log session to a file
pu*sh	Push to a shell
qui*t	Exit kermit
rec*eive	Recieve a file from the remote kermit
ren*ame	Rename a file (OS-9 command)

rem*ote	Execute a remote command, see below
sen*d	Send a file to the remote kermit
set b*lock-check	Select checksum (1) or CRC (2) block check
set d*ebug-level	Set the level of debug output
spa*ce	Display the amount of free diskspace (OS-9 free command)
take	Take a file from the remote server, not implemented
typ*e	List the contents of a file (OS-9 list command)

One of the most useful of the above commands is kermit's remote command, which allows you to query the remote kermit server. You MUST have the remote kermit in server mode to use the remote commands. Not all versions of kermit support the remote commands or all remote commands. All of the below commands are run on the remote machine, with all output being sent through kermit to your screen.

com*mand	Execute the given command
cop*y	Copy file(s)
cw*d	Change the current directory
de*lete	Delete file(s)
di*rectory	List a directory
l*ogin	Connect to a dedicated kermit file server
r*ename	Rename file(s)
s*pace	Display the amount of free diskspace
t*ype	List file(s)

- CREDITS -

- NOTES -